



CPSC203 – Introduction to Problem Solving and Using Application Software

Fall 2009

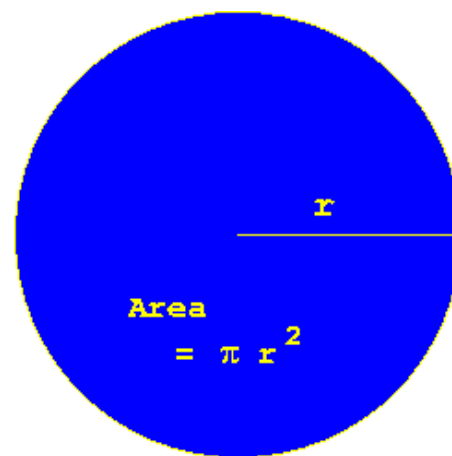
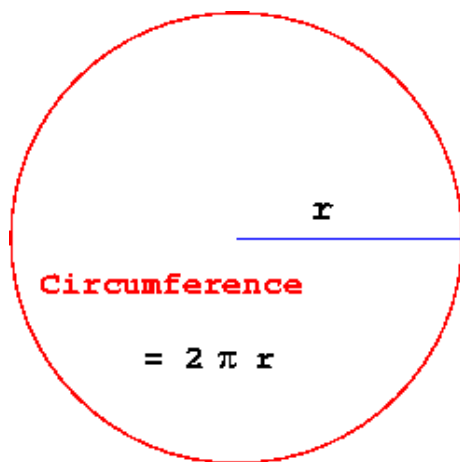
Tutorial 25, Mehrdad Nurolahzade

Introduction

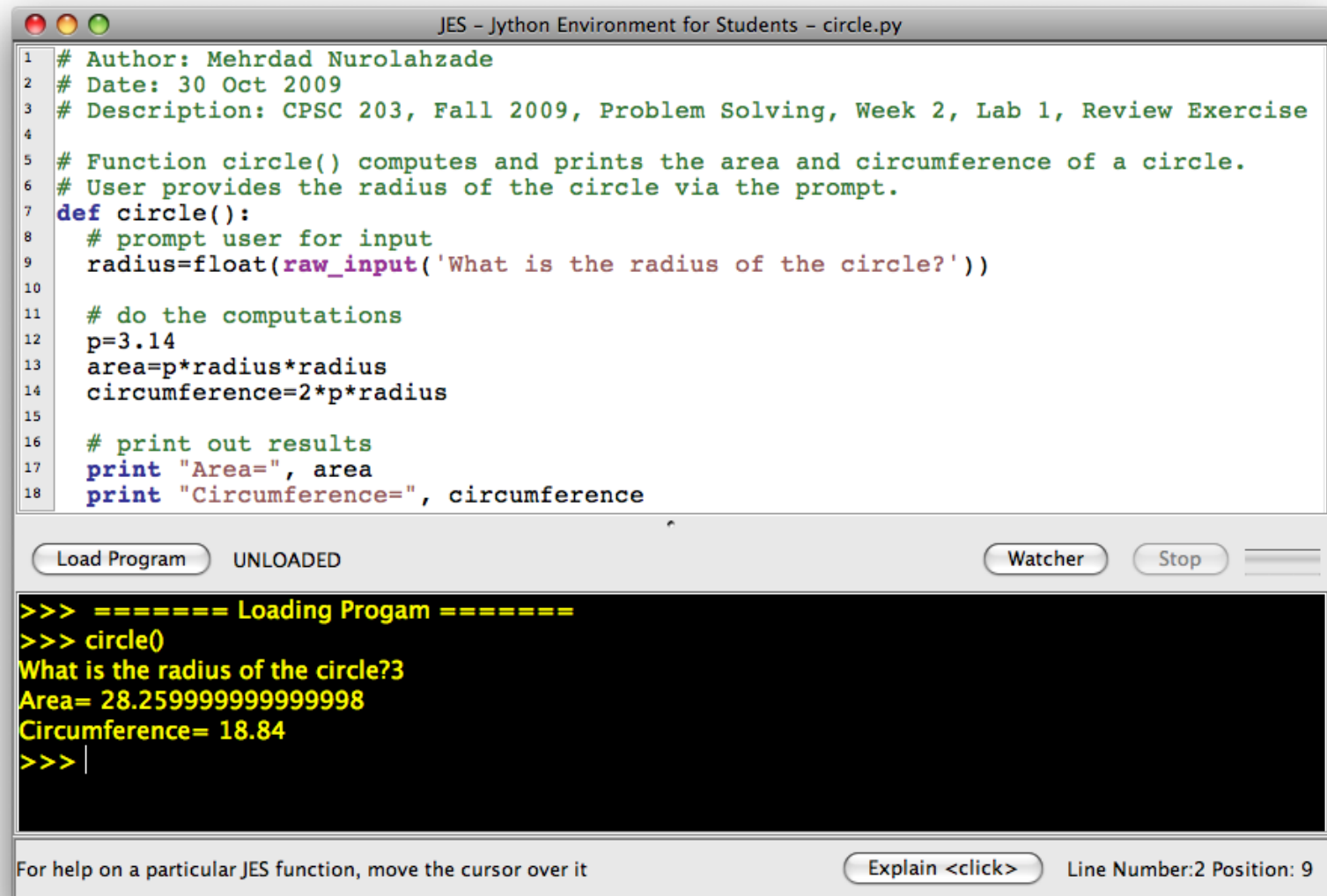
- Jython Basics Review Exercise
- Using Loops
- Using Conditions

Circle Area and Circumference (1)

- Write a function named **circle()** that asks the user to enter the radius of a circle, then computes and prints out the area and circumferences of that circle.



Circle Area and Circumference (2)



```
1 # Author: Mehrdad Nurolahzade
2 # Date: 30 Oct 2009
3 # Description: CPSC 203, Fall 2009, Problem Solving, Week 2, Lab 1, Review Exercise
4
5 # Function circle() computes and prints the area and circumference of a circle.
6 # User provides the radius of the circle via the prompt.
7 def circle():
8     # prompt user for input
9     radius=float(raw_input('What is the radius of the circle?'))
10
11     # do the computations
12     p=3.14
13     area=p*radius*radius
14     circumference=2*p*radius
15
16     # print out results
17     print "Area=", area
18     print "Circumference=", circumference
```

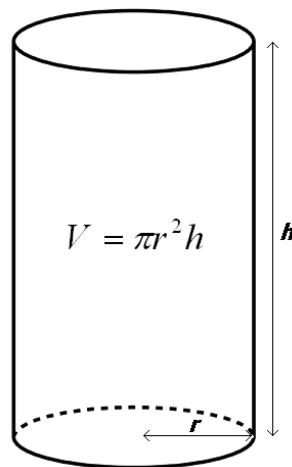
Load Program UNLOADED Watcher Stop

```
>>> ===== Loading Program =====
>>> circle()
What is the radius of the circle?3
Area= 28.259999999999998
Circumference= 18.84
>>> |
```

For help on a particular JES function, move the cursor over it Explain <click> Line Number:2 Position: 9

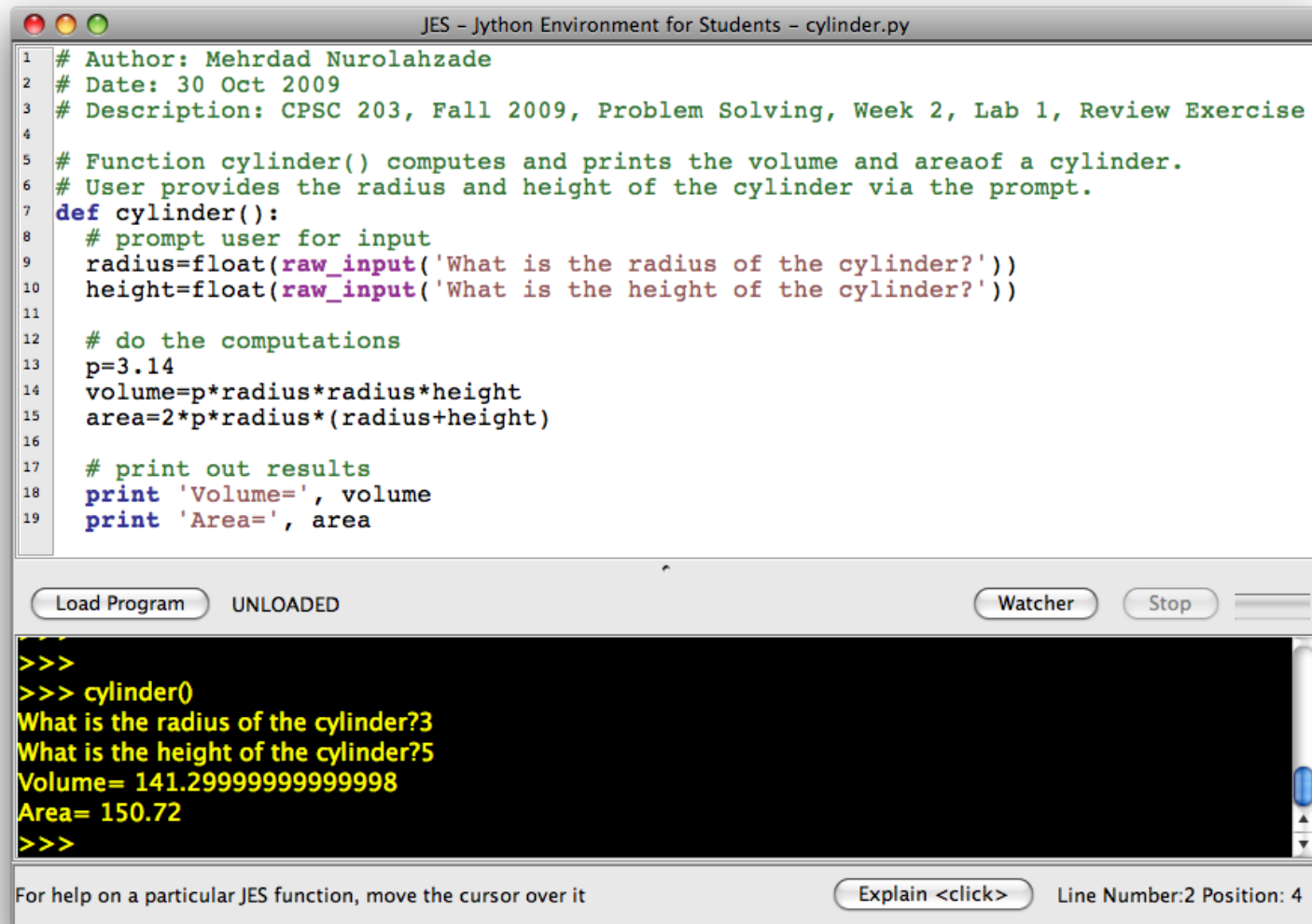
Cylinder Volume and Area (1)

- Write a function named **cylinder()** that asks the user to enter the radius and height of a cylinder, then computes and prints out the volume and area of that cylinder.



$$A = 2\pi r^2 + 2\pi r h = 2\pi r(r + h).$$

Cylinder Volume and Area (2)



```
1 # Author: Mehrdad Nurolahzade
2 # Date: 30 Oct 2009
3 # Description: CPSC 203, Fall 2009, Problem Solving, Week 2, Lab 1, Review Exercise
4
5 # Function cylinder() computes and prints the volume and area of a cylinder.
6 # User provides the radius and height of the cylinder via the prompt.
7 def cylinder():
8     # prompt user for input
9     radius=float(raw_input('What is the radius of the cylinder?'))
10    height=float(raw_input('What is the height of the cylinder?'))
11
12    # do the computations
13    p=3.14
14    volume=p*radius*radius*height
15    area=2*p*radius*(radius+height)
16
17    # print out results
18    print 'Volume=', volume
19    print 'Area=', area
```

Load Program UNLOADED Watcher Stop

```
>>>
>>> cylinder()
What is the radius of the cylinder?3
What is the height of the cylinder?5
Volume= 141.29999999999998
Area= 150.72
>>>
```

For help on a particular JES function, move the cursor over it Explain <click> Line Number:2 Position: 4

if Statement

- The **if** statement selects actions to perform.

- General syntax:

```
if if-condition:  
    if-body
```

- Example:

```
radius=float(raw_input('Enter circle radius'))  
if radius<0:  
    print "Circle radius has to be positive!"
```

Comparison Operators

- Different operators can be used in the condition of an **if** statement:

- == (equal)
- != (not equal)
- < (less than)
- <= (less than or equal)
- > (greater than)
- >= (greater than or equal)

```
if instructor=='Jalal':  
    print 'You are in L03 or  
    L04'
```

```
if instructor!='Jalal':  
    print 'You are in L01 or  
    L02'
```

```
if age>13:  
    if age<20:  
        print 'You are a teen'
```


if-else Statement

- An **if** statement can have an **else** part.
- General syntax of an **if-else** statement:

```
if if-condition:
```

```
    if-body
```

```
else:
```

```
    else-body
```

- **Example:**

```
if sex=='Male':
```

```
    print 'Good morning Mr. '+name
```

```
else:
```

```
    print 'Good morning Mrs. '+name
```

Nested if-else Statements

- The **if** statement may contain other **if** statements.

```
if grade >= 90:
    letter_grade = 'A'
else:
    if grade >= 80:
        letter_grade = 'B'
    else:
        if grade >= 70:
            letter_grade = 'C'
        else:
            if grade >= 60:
                letter_grade = 'D'
            else:
                if grade >= 50:
                    letter_grade = 'E'
                else:
                    letter_grade = 'F'
```

```
if grade >= 90:
    letter_grade = 'A'
elif grade >= 80:
    letter_grade = 'B'
elif grade >= 70:
    letter_grade = 'C'
elif grade >= 60:
    letter_grade = 'D'
elif grade >= 50:
    letter_grade = 'E'
else:
    letter_grade = 'F'
```

Logic Operators

- More complex logical conditions can be built using **and**, **or**, **not** operators.

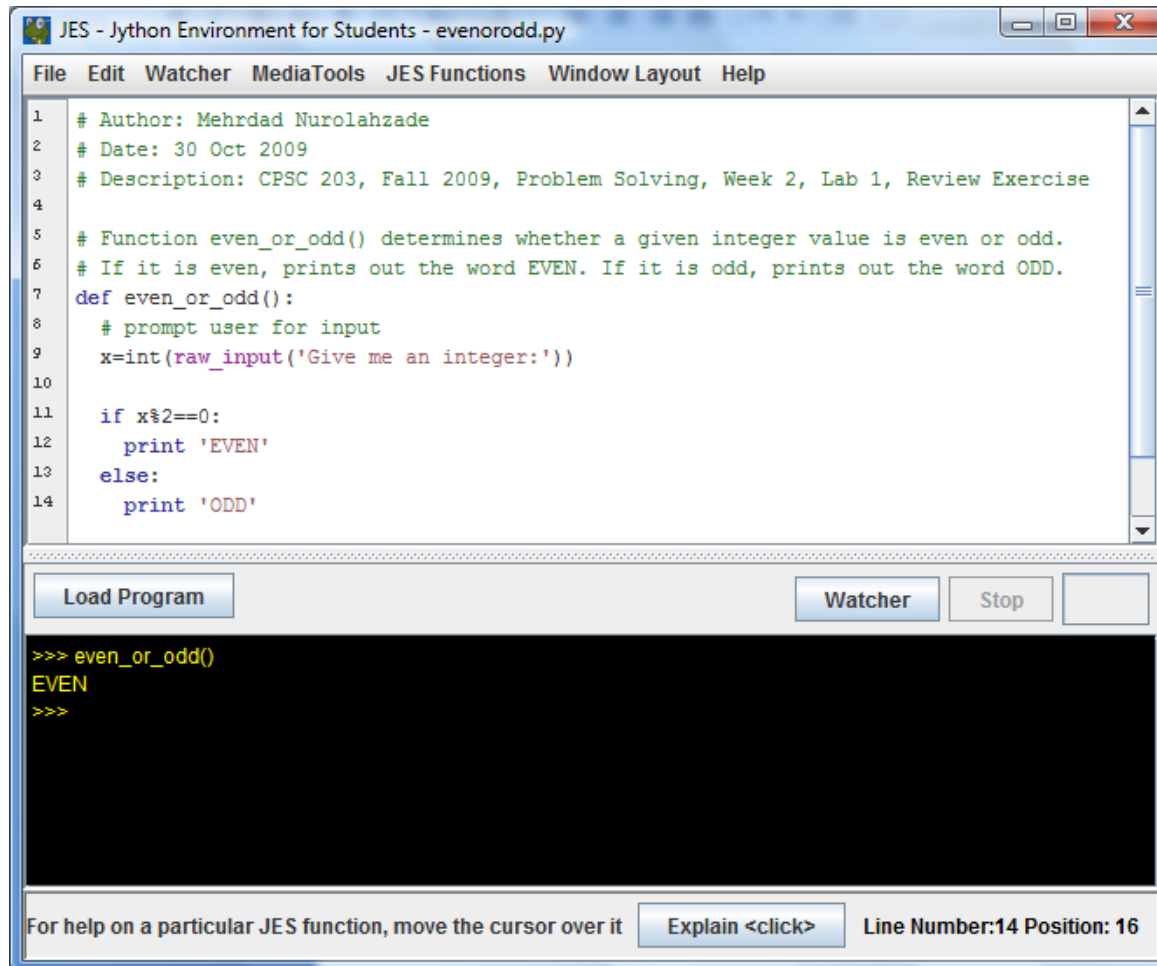
```
if day=='Monday' and hour>18:
    print 'Pizza'
elif day=='Tuesday' or day=='Thursday':
    print 'Wings'
elif hour>19:
    print 'Hamburger'
else:
    print 'Pasta'
```

Logic Exercise (1)

- Write a function **even_or_odd()** that asks user to enter an integer value, then prints out EVEN or ODD if the given value is even or odd respectively.
- **Example runs:**
Give me an integer: 3
ODD

Give me an integer: 6
EVEN

Logic Exercise (2)



The screenshot shows the JES IDE window titled "JES - Jython Environment for Students - evenorodd.py". The menu bar includes File, Edit, Watcher, MediaTools, JES Functions, Window Layout, and Help. The editor displays a Python script with the following code:

```
1 # Author: Mehrdad Nurolahzade
2 # Date: 30 Oct 2009
3 # Description: CPSC 203, Fall 2009, Problem Solving, Week 2, Lab 1, Review Exercise
4
5 # Function even_or_odd() determines whether a given integer value is even or odd.
6 # If it is even, prints out the word EVEN. If it is odd, prints out the word ODD.
7 def even_or_odd():
8     # prompt user for input
9     x=int(raw_input('Give me an integer:'))
10
11     if x%2==0:
12         print 'EVEN'
13     else:
14         print 'ODD'
```

Below the editor, there are buttons for "Load Program", "Watcher", "Stop", and a blank button. The execution area shows the following output:

```
>>> even_or_odd()
EVEN
>>>
```

At the bottom, a status bar indicates "For help on a particular JES function, move the cursor over it" with an "Explain <click>" button, and "Line Number:14 Position: 16".

Logic Exercise (3)

- Write a function named **salary_tax()** that asks user to enter his/her own monthly salary and computes and prints out the salary tax. If salary is below \$3,000 the tax is zero. If salary is between \$3,000 and \$4,000 the tax is 5%. If salary is between \$4,000 and \$5,000 the tax is 10%. If salary is above \$5,000 the tax is 15%.

- Example run:

```
Enter your monthly salary: 4200
```

```
You are monthly tax is: $420.0
```

Lists

- A list is an ordered set of indexed elements.

```
names=[ 'John', 'Mike', 'Rose', 'James', 'Tina']  
numbers=[3, 5, 1, 6, 2, 5]
```

- Elements are numbered left to right.
- The index of the first element is 0.
- The index of elements is used to reference them:

| | |
|-------------------------|----------------|
| <code>names[1]</code> | that is 'Mike' |
| <code>numbers[3]</code> | that is 6 |
| <code>numbers[0]</code> | that is 3 |

List Related Functions (1)

```
# Author: Mehrdad Nurolahzade
# Date: 30 Oct 2009
# Description: CPSC 203, Fall 2009, Problem Solving, Week 2, Lab 1, Review Exercise

# Function list_example() demonstrates some useful list related functions
def list_example():
    list1=['a', 'g', 'b', 'a', 'd', 'b']
    list2=['d', 'a', 'c']
    # len(list) returns the number of elements in list
    print "Length of list1 is", len(list1)
    # list.index(element) returns the index of the first occurrence of element in list
    print "Index of 'b' in list1 is", list1.index('b')
    # list.append(element) adds element to the end of list
    list1.append('c')
    # list.insert(i, element) adds element to list at position i
    list2.insert(0, 'b')
    # creates a new list by concatenating list1 and list2
    list3=list1+list2
    # list1.extend(list2) appends list2 to list1
    list1.extend(list2)
```

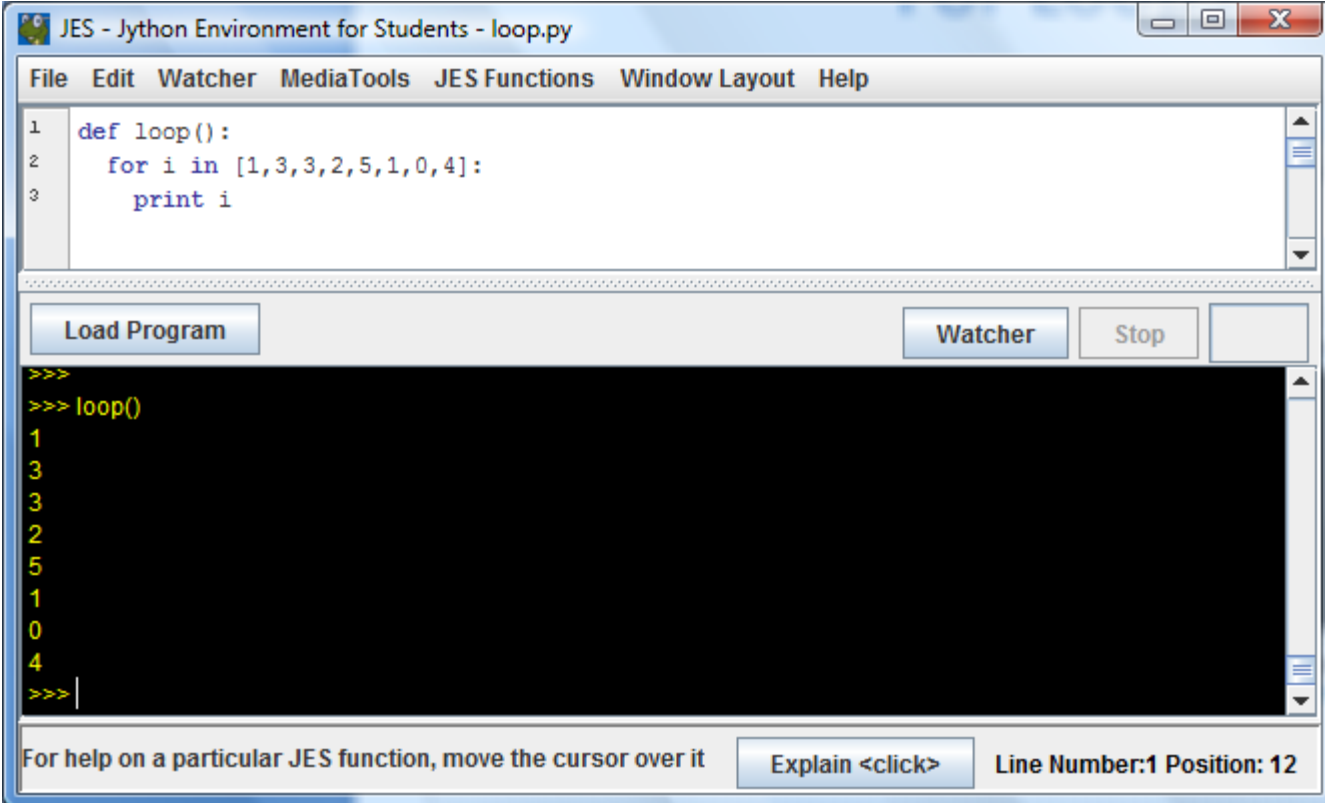

List Related Functions (2)

```
# list=[e]*n initializes list to n elements of value e
list4=[0]*5
# list.remove(element) deletes the first occurrence of element in list
list1.remove('g')
# list.pop() deletes and returns the last element in list
last=list1.pop()
# list.sort() sorts list in ascending order
list1.sort()
# list.reverse() reverses list
list1.reverse()
# min(list) returns the minimum element in list
minimum=min(list1)
# max(list) returns the maximum element in list
maximum=max(list1)
# element in list returns true if element is in list, false otherwise
if 'a' in list1:
    print "a is in list1"
```

Loops

- Loops are programming structures that allow us to repeat some statements as many times as required.
- For loops are suitable when you need to repeat something for a known number of times.
- While loops are used when we need to repeat something until a certain condition is met.

for Loop (1)



The screenshot shows the JES IDE window titled "JES - Jython Environment for Students - loop.py". The menu bar includes File, Edit, Watcher, MediaTools, JES Functions, Window Layout, and Help. The editor displays the following Python code:

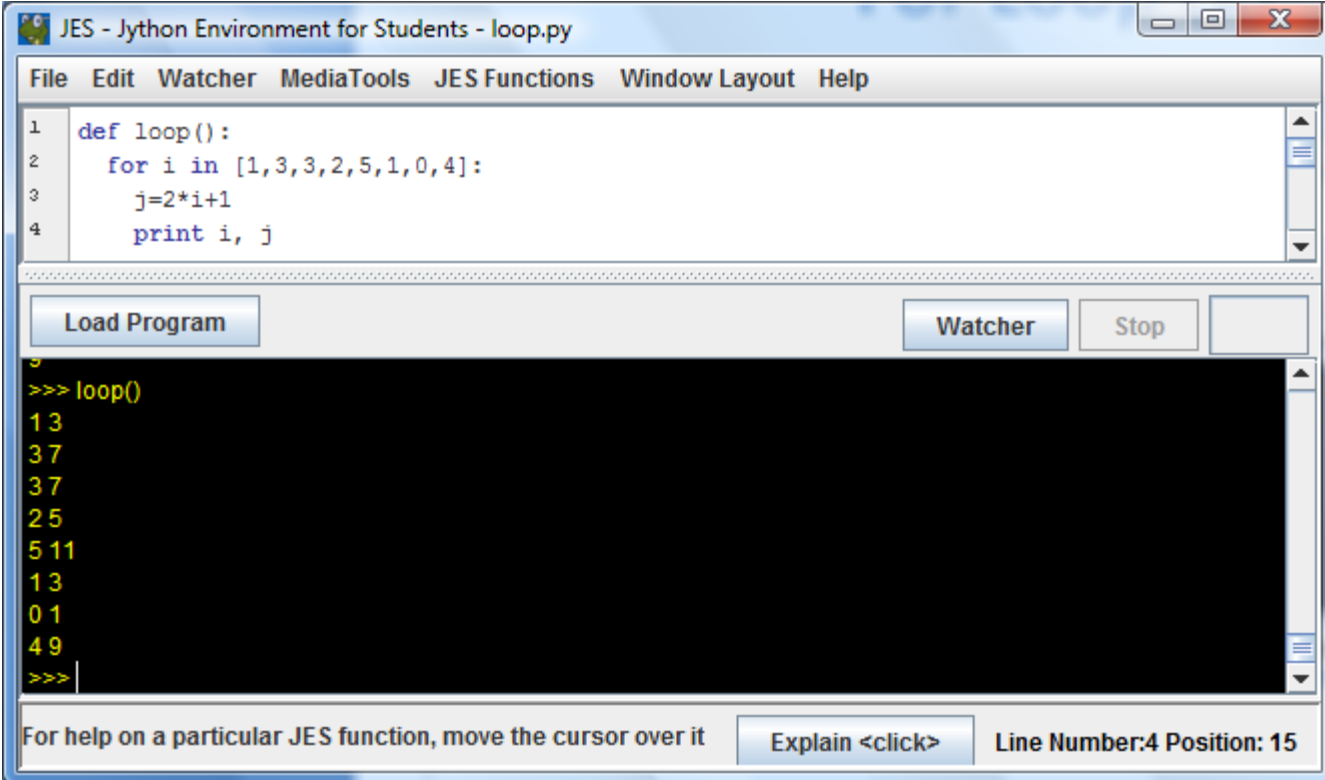
```
1 def loop():  
2     for i in [1,3,3,2,5,1,0,4]:  
3         print i
```

Below the editor is a toolbar with buttons for "Load Program", "Watcher", "Stop", and a blank button. The output window shows the execution of the code:

```
>>>  
>>> loop()  
1  
3  
3  
2  
5  
1  
0  
4  
>>> |
```

At the bottom, a status bar provides help information: "For help on a particular JES function, move the cursor over it" with an "Explain <click>" button, and "Line Number:1 Position: 12".

for Loop (2)



The screenshot shows the JES (Jython Environment for Students) interface. The top window displays a Python script named `loop.py` with the following code:

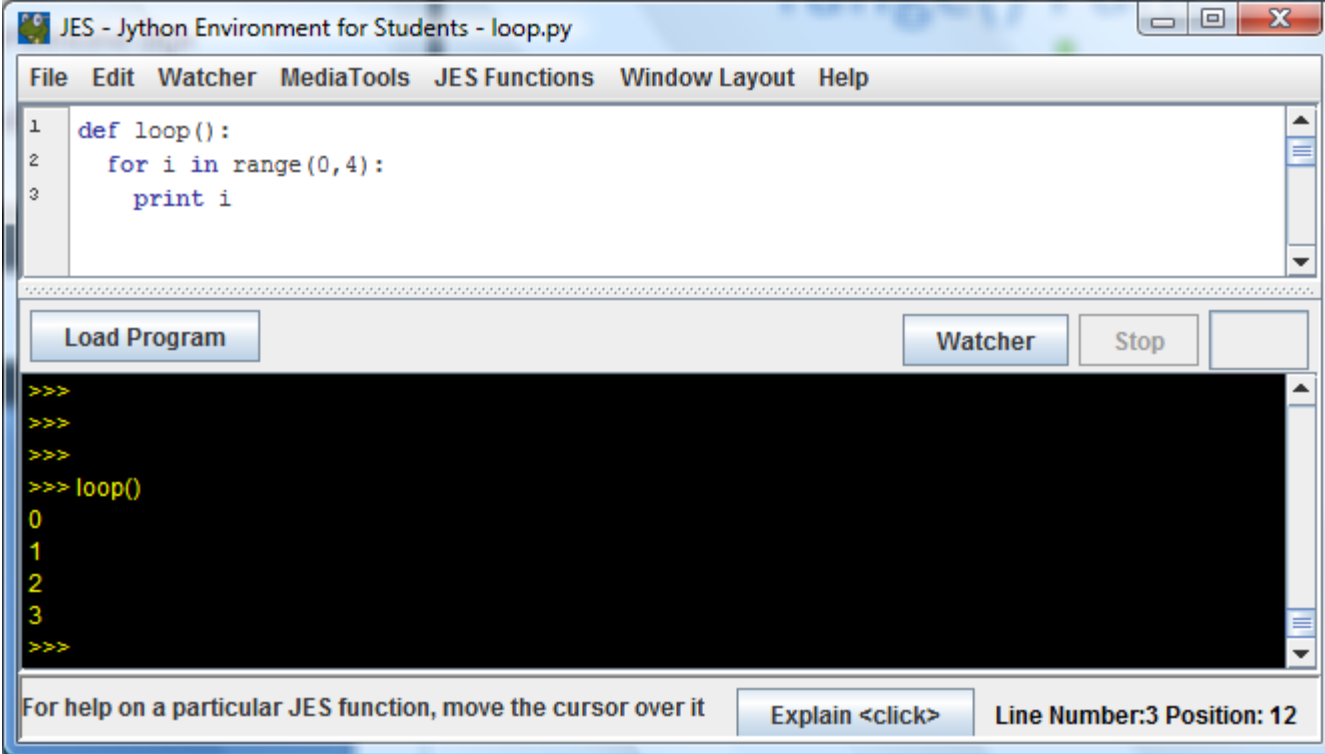
```
1 def loop():  
2     for i in [1,3,3,2,5,1,0,4]:  
3         j=2*i+1  
4         print i, j
```

Below the code editor, there are buttons for `Load Program`, `Watcher`, and `Stop`. The bottom window shows the output of the program, which is the result of calling `loop()`:

```
>>> loop()  
1 3  
3 7  
3 7  
2 5  
5 11  
1 3  
0 1  
4 9  
>>>
```

At the bottom of the interface, there is a status bar that reads: "For help on a particular JES function, move the cursor over it Explain <click> Line Number:4 Position: 15".

range() Function (1)



The screenshot shows the JES (Jython Environment for Students) window. The title bar reads "JES - Jython Environment for Students - loop.py". The menu bar includes "File", "Edit", "Watcher", "MediaTools", "JES Functions", "Window Layout", and "Help". The main editor area contains the following Python code:

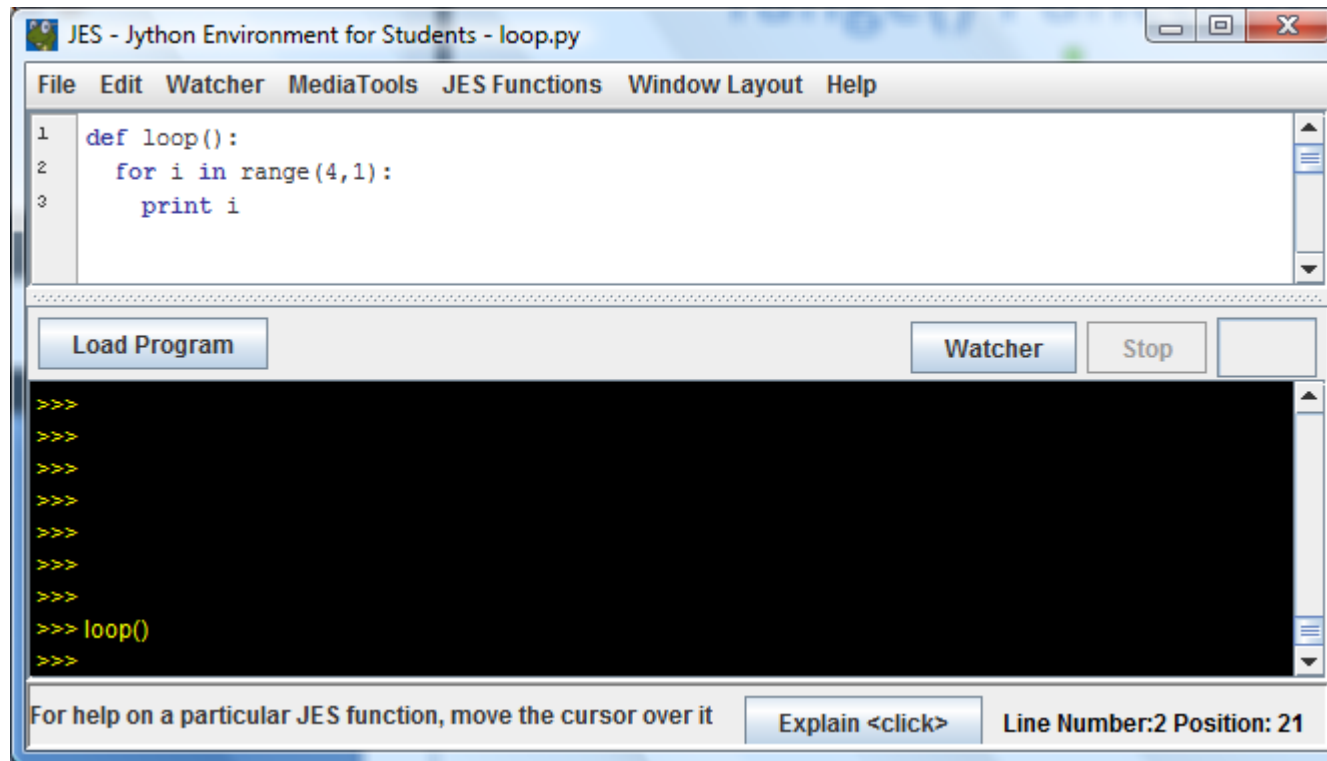
```
1 def loop():  
2     for i in range(0,4):  
3         print i
```

Below the editor is a toolbar with buttons for "Load Program", "Watcher", and "Stop". The console area, which has a black background, shows the execution of the program:

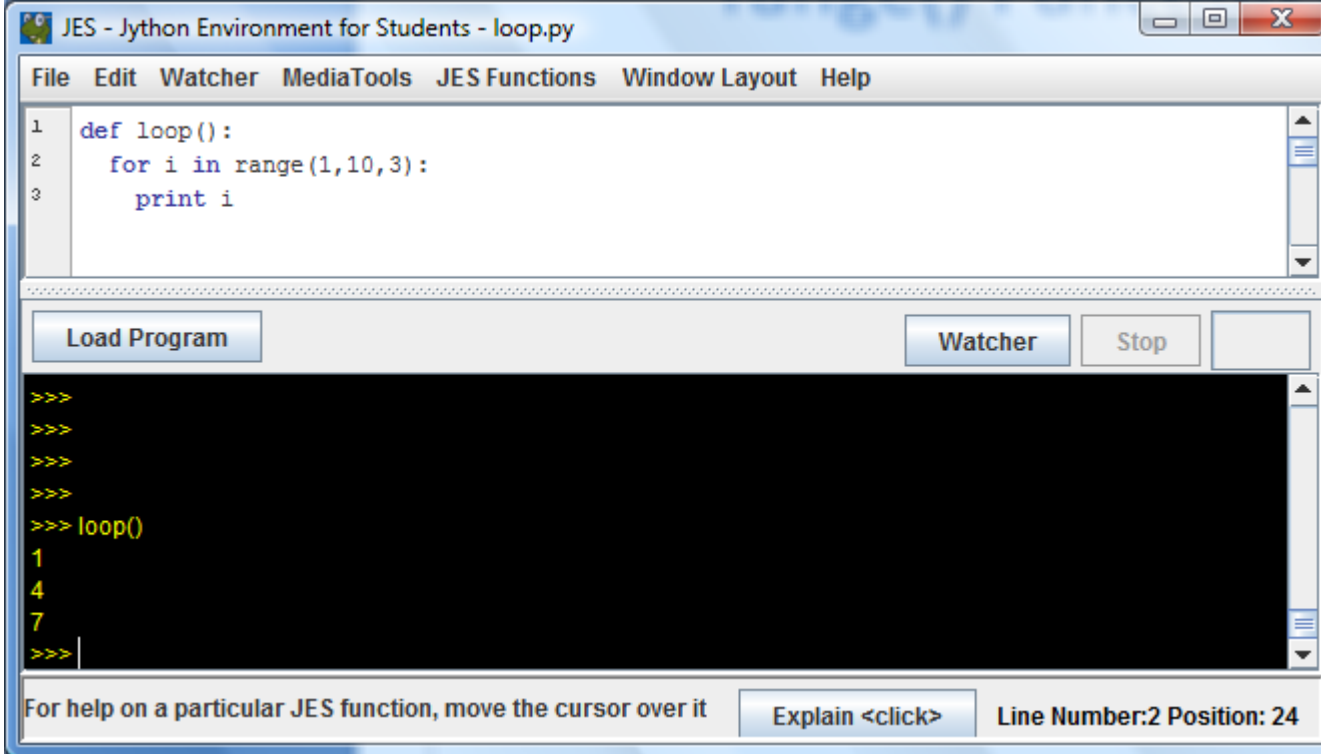
```
>>>  
>>>  
>>>  
>>> loop()  
0  
1  
2  
3  
>>>
```

At the bottom of the window, there is a status bar that says "For help on a particular JES function, move the cursor over it" followed by an "Explain <click>" button and "Line Number:3 Position: 12".

range() Function (2)



range() Function (3)



The screenshot shows the JES (Jython Environment for Students) interface. The top window displays a Python script named `loop.py` with the following code:

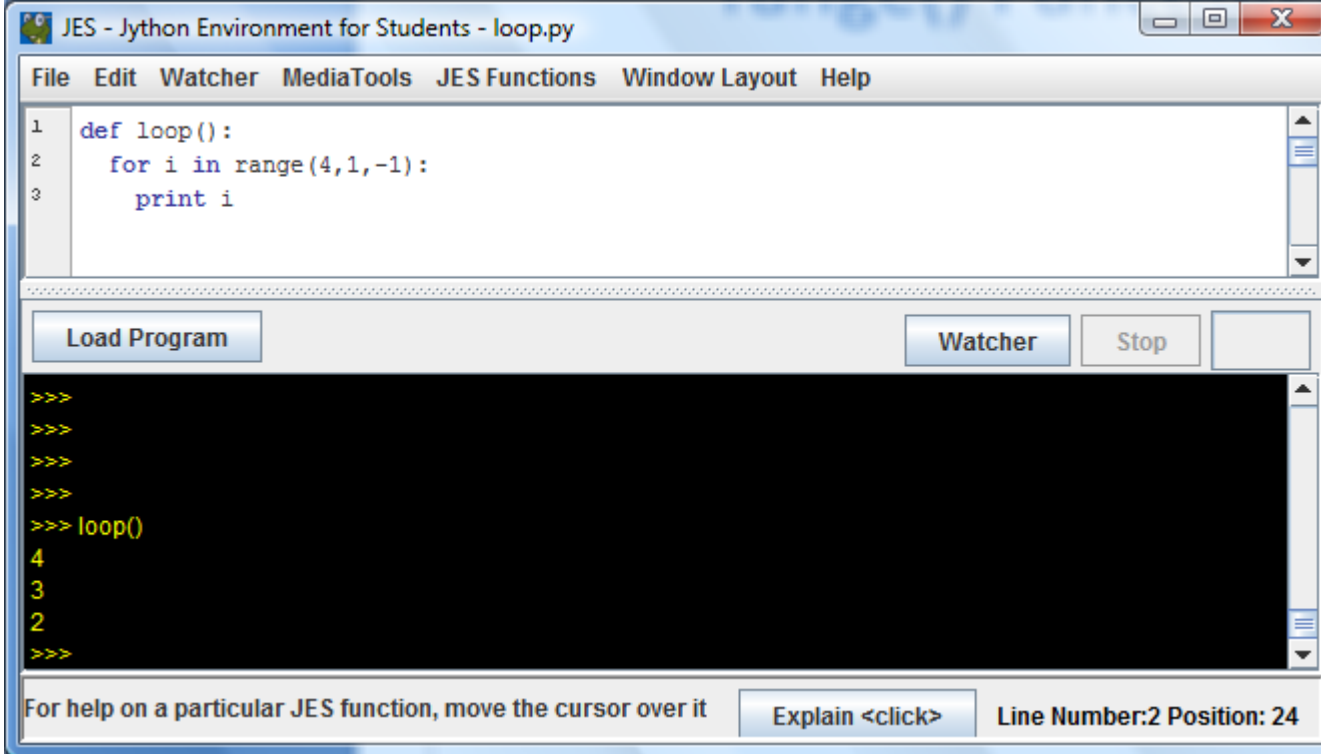
```
1 def loop():  
2     for i in range(1,10,3):  
3         print i
```

Below the code editor is a toolbar with buttons for `Load Program`, `Watcher`, and `Stop`. The bottom window shows the interactive prompt with the following input and output:

```
>>>  
>>>  
>>>  
>>>  
>>> loop()  
1  
4  
7  
>>>
```

At the bottom of the interface, there is a status bar that reads: "For help on a particular JES function, move the cursor over it Explain <click> Line Number:2 Position: 24".

range() Function (4)



The screenshot shows the JES (Jython Environment for Students) interface. The top window, titled "JES - Jython Environment for Students - loop.py", contains a Python script:

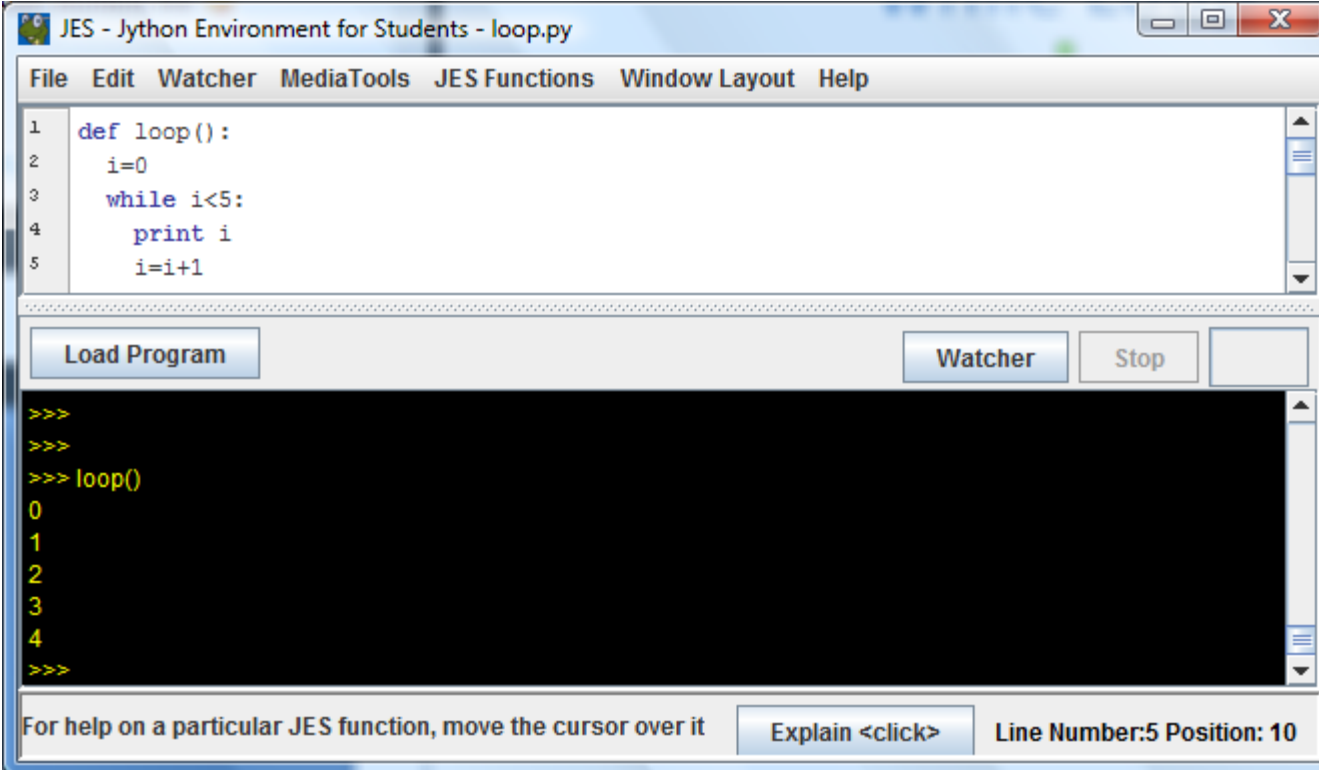
```
1 def loop():  
2     for i in range(4,1,-1):  
3         print i
```

Below the editor are buttons for "Load Program", "Watcher", "Stop", and a "Run" button (represented by a play icon). The bottom pane shows the interactive prompt with the following input and output:

```
>>>  
>>>  
>>>  
>>>  
>>> loop()  
4  
3  
2  
>>>
```

At the bottom of the window, a status bar indicates "For help on a particular JES function, move the cursor over it" with an "Explain <click>" button, and shows "Line Number:2 Position: 24".

while Loop (1)



The screenshot shows the JES IDE window titled "JES - Jython Environment for Students - loop.py". The menu bar includes File, Edit, Watcher, MediaTools, JES Functions, Window Layout, and Help. The editor displays a Python function:

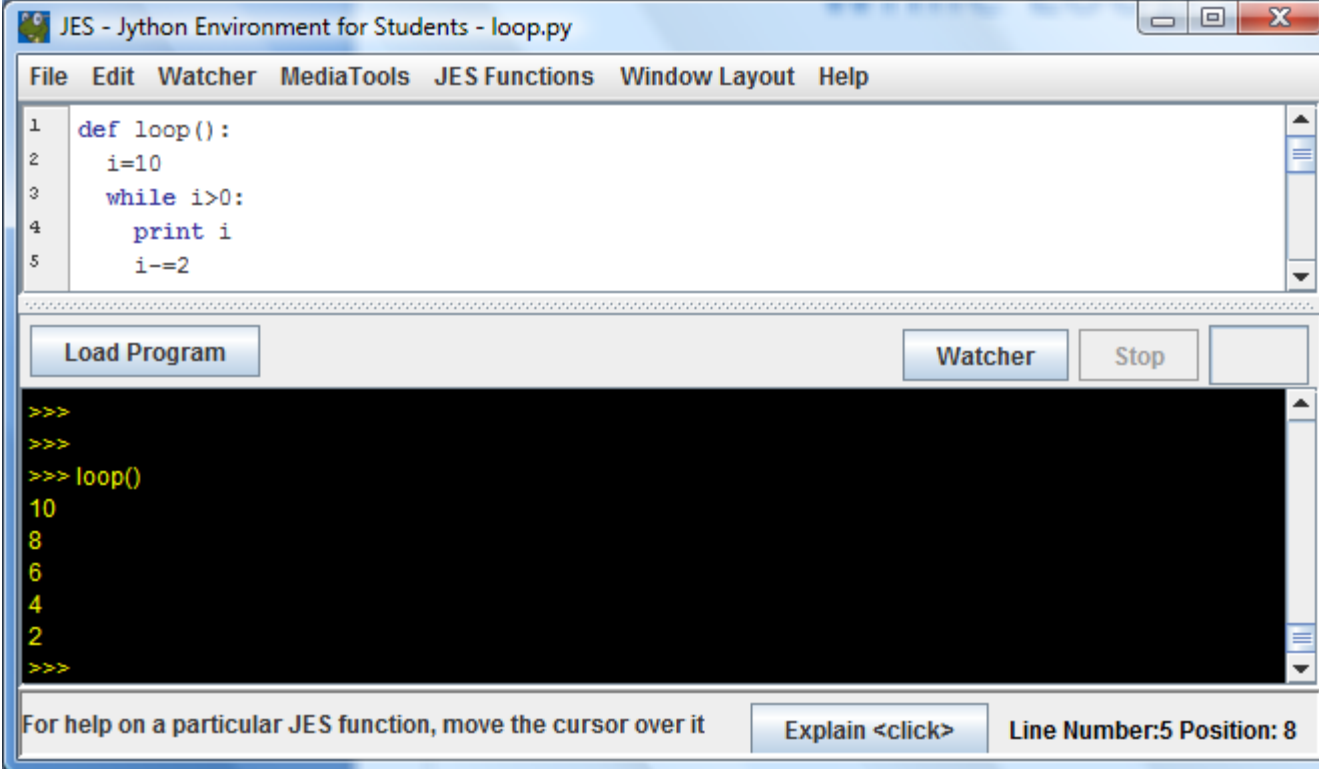
```
1 def loop():
2     i=0
3     while i<5:
4         print i
5         i=i+1
```

Below the editor is a toolbar with buttons for "Load Program", "Watcher", "Stop", and a blank button. The console area shows the execution of the function:

```
>>>
>>>
>>> loop()
0
1
2
3
4
>>>
```

At the bottom, a status bar provides help information and the current cursor position: "For help on a particular JES function, move the cursor over it", "Explain <click>", and "Line Number:5 Position: 10".

while Loop (2)



The screenshot shows the JES (Jython Environment for Students) interface. The top window displays a Python script named `loop.py` with the following code:

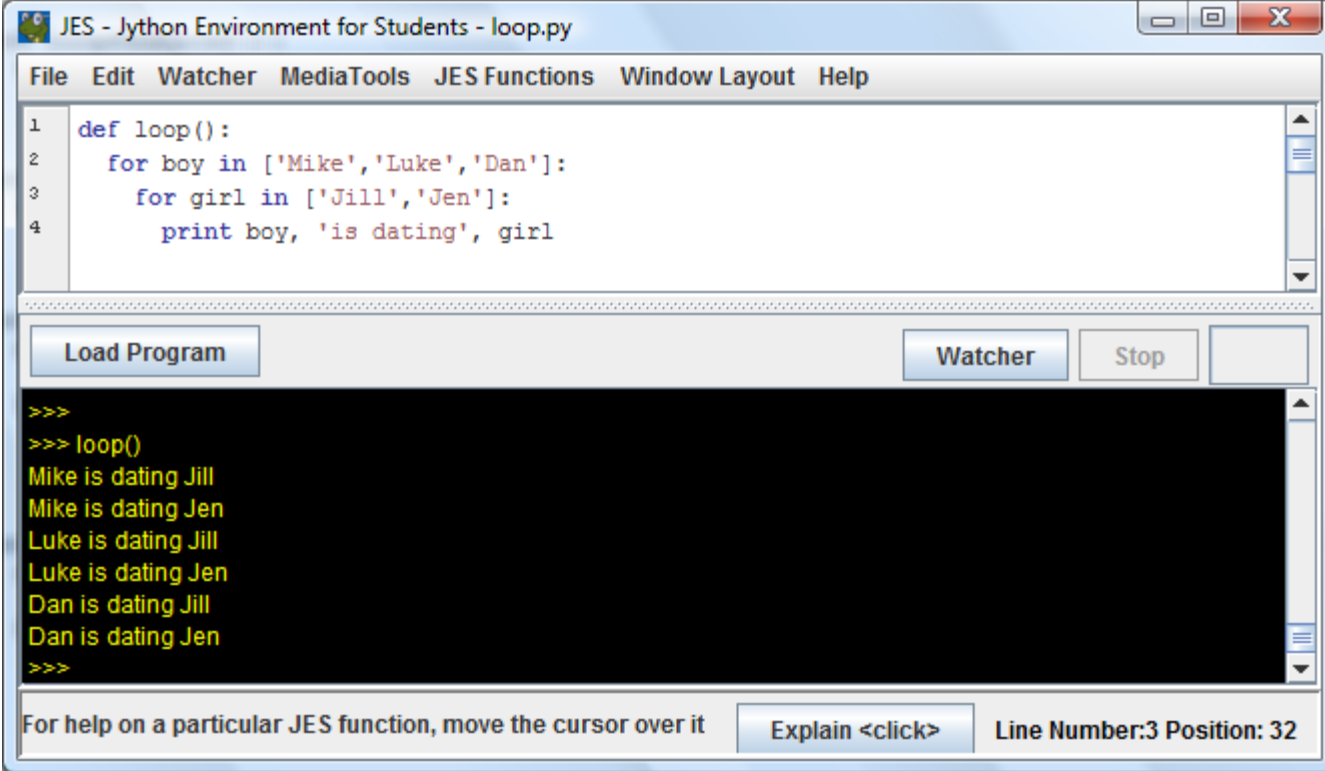
```
1 def loop():  
2     i=10  
3     while i>0:  
4         print i  
5         i-=2
```

Below the code editor, there are buttons for `Load Program`, `Watcher`, and `Stop`. The `Watcher` button is active, showing the output of the program in a black console window:

```
>>>  
>>>  
>>> loop()  
10  
8  
6  
4  
2  
>>>
```

At the bottom of the interface, there is a status bar with the text "For help on a particular JES function, move the cursor over it" and a button labeled "Explain <click>". The status bar also displays "Line Number:5 Position: 8".

Nested Loops



The screenshot shows the JES IDE window titled "JES - Jython Environment for Students - loop.py". The menu bar includes File, Edit, Watcher, MediaTools, JES Functions, Window Layout, and Help. The code editor contains the following Python code:

```
1 def loop():
2     for boy in ['Mike', 'Luke', 'Dan']:
3         for girl in ['Jill', 'Jen']:
4             print boy, 'is dating', girl
```

Below the code editor is a toolbar with buttons for "Load Program", "Watcher", "Stop", and an empty button. The output window, which has a black background, shows the execution results:

```
>>>
>>> loop()
Mike is dating Jill
Mike is dating Jen
Luke is dating Jill
Luke is dating Jen
Dan is dating Jill
Dan is dating Jen
>>>
```

At the bottom of the window, a status bar displays the text "For help on a particular JES function, move the cursor over it" next to an "Explain <click>" button, and "Line Number:3 Position: 32" on the right.

Loop Exercise (1)

- Write a function named **sum()** that asks user to enter a number N and then calculates and prints out the sum of values from 1 to N.

- Example run:

```
Give me a value for N: 5  
Sum of values 1 to N is 15
```

Loop Exercise (2)

- Write a function named **list_sum()** that defines two same size lists named list1 and list2 with values [1, 3, 2, 3, 4] and [0, 1, 2, 0, 3] respectively. The function should create and print out a third list named list3 whose elements are the sum of same position elements in list1 and list2.
- Example run:
`list3= [1, 4, 4, 3, 7]`